GETMAX / FIELD PLAYBOOK

The free 24/7 server playbook
Oracle Cloud Always Free

How to get a real Linux server with a public IP that costs nothing and never expires — including the "Out of capacity" problem everyone hits on the ARM machines, and the retry-script strategy that gets around it. Written from running one, not from reading about it.

$0	2 + 1	10 TB	200 GB
FOREVER, IF YOU NEVER UPGRADE	2 AMD MICROS + 1 ARM BOX (2 OCPU/12GB)	FREE EGRESS PER MONTH	BLOCK STORAGE

AUTHOR Sriram Raghavan · GetMax Healthcare Solutions

DATE July 2026 · specs verified against Oracle documentation, incl. the June 15, 2026 ARM reduction

WEB sriram.getmaxglobal.com/blog/oracle-free-server

01 What "Always Free" actually is

An Oracle Cloud account gives you two separate things, and mixing them up is the first mistake people make.

- **The 30-day trial.** $300 of credits, valid for 30 days, usable on almost anything. This part expires. Anything built purely on trial credits is reclaimed when the trial ends.
- **Always Free resources.** A fixed allowance of compute, storage, databases and networking that never expires. Oracle's documentation states it plainly: "After your trial ends, your account remains active. There is no interruption to the availability of the Always Free Resources you have provisioned."

Everything in this playbook uses only the Always Free allowance. Skip the trial credits entirely if you want zero risk of building on something that disappears on day 31.

02 The allowance — verified July 2026

RESOURCE	WHAT YOU GET
AMD compute	2 × VM.Standard.E2.1.Micro instances — 1/8 OCPU and 1 GB RAM each. Launch instantly in most regions.
ARM compute	Ampere A1: 1,500 OCPU-hours + 9,000 GB-hours per month. That equals one 2 OCPU / 12 GB machine running 24/7, or the same total split across smaller instances.
Block storage	200 GB total (boot volumes count against this) + 5 volume backups.
Object storage	20 GB, 50,000 API requests per month.
Databases	2 × Autonomous Database at 20 GB each · NoSQL: 3 tables × 25 GB · MySQL HeatWave 50 GB.
Outbound transfer	10 TB per month — more egress than most VPS plans at any price.
Load balancer	1 flexible load balancer, 10 Mbps.
Email delivery	3,000 emails per month.

> **THE JUNE 2026 CUT — MOST GUIDES ARE NOW WRONG**
>
> The ARM allowance was 4 OCPUs / 24 GB for years. On **June 15, 2026** Oracle cut it to **2 OCPUs / 12 GB** — quietly, via a documentation update, with no announcement. Guides promising a free 4-core 24 GB machine are out of date. A related caveat from the change: if you terminate an existing ARM instance that's above the new limit, you may not be able to recreate it at the old size.

03 Signing up — the three gotchas

Gotcha 1: your home region is permanent

Always Free compute and databases can only be created in the **home region** you choose at signup, and it can never be changed. Pick the region closest to your users. There is no reliable way to know in advance which region has spare ARM capacity, so optimize for latency and accept the capacity fight wherever you land. We picked Hyderabad (ap-hyderabad-1) and fought the fight there.

Gotcha 2: the card is for verification, not billing

Signup at `signup.oraclecloud.com` requires a mobile number and a credit or debit card. Oracle places a small temporary authorization hold and releases it. Their words: "Your credit card will not be charged unless you upgrade your account." Some debit cards (Indian ones especially) fail verification — if that happens, use a credit card. Expect the identity checks to be stricter than AWS; mismatched name/address details are the most common rejection reason.

Gotcha 3: never upgrade

The only path from this setup to a bill is converting the account to Pay As You Go. Don't do it — not for "more capacity", not because a console banner suggests it. On a free account, hitting a limit means a request fails; on an upgraded account, it means you pay. The free account physically cannot bill you.

04 Day one: launch the AMD micro (it works immediately)

The two AMD micro instances are a **separate allowance** from the ARM machines, and in our experience they launch on the first attempt — ours came up in about a minute while the ARM request had been failing for days. Take one immediately so you have a working server today.

- Console → Compute → Instances → Create instance.
- Image: Ubuntu 22.04 (or 24.04). Shape: **VM.Standard.E2.1.Micro** — it's tagged "Always Free-eligible".
- Networking: create a VCN with a public subnet if you don't have one; assign a public IPv4 address.
- SSH keys: upload your public key (generate one with `ssh-keygen -t ed25519` if needed). There is no password login — lose the key, lose the box.
- Create. You'll have a public IP in under two minutes.

It's a small machine — 1/8 of a CPU core and 1 GB of RAM — but it's a real always-on Linux box with a public IP and 10 TB of monthly egress, at $0.

05 The ARM hunt: "Out of capacity" and the retry script

The Ampere A1 machines are what everyone actually wants, so in most regions free-tier requests meet **"Out of host capacity"** — over and over, for days or weeks. This is normal. Free requests get whatever is left after paying customers, and in busy regions that's usually nothing. Clicking retry in the console is a waste of your life. Script it.

> **THE STRATEGY, FROM OUR OWN LOOP**
>
> Call the launch API on a timer — **every ~12 minutes**, around the clock — trying the **smallest ARM shape first** (1 OCPU / 6 GB), then stepping up. Small requests fit into capacity gaps that a full 2/12 request won't. Once anything lands, resize it upward in place later — same instance, same IP. Our loop has logged every attempt in Hyderabad as a 500/429 "no capacity" for over a week and it just keeps going. That's the game: patience, automated.

The skeleton, using Oracle's Python SDK (`pip install oci`; auth via an API key in `~/.oci/config`):

```
import oci, time, itertools

cfg = oci.config.from_file()          # ~/.oci/config
compute = oci.core.ComputeClient(cfg)
SHAPES = [(1, 6), (2, 12)]            # (ocpus, ram_gb) — smallest first

def try_launch(ocpus, ram_gb):
    details = oci.core.models.LaunchInstanceDetails(
        compartment_id=cfg["tenancy"],
        availability_domain=AD,        # loop over ADs if your region has several
        shape="VM.Standard.A1.Flex",
        shape_config=oci.core.models.LaunchInstanceShapeConfigDetails(
            ocpus=ocpus, memory_in_gbs=ram_gb),
        source_details=IMAGE,          # Ubuntu 22.04 aarch64
        create_vnic_details=VNIC,      # public subnet, assign public IP
        metadata={"ssh_authorized_keys": PUBKEY})
    return compute.launch_instance(details)

while True:
    for ocpus, ram in SHAPES:
        try:
            inst = try_launch(ocpus, ram)
            print("LANDED:", inst.data.id); raise SystemExit
        except oci.exceptions.ServiceError as e:
            if e.status in (429, 500): pass    # out of capacity — expected
            else: raise                        # a real error — fix it, don't loop it
    time.sleep(12 * 60)
```

- **Keep the interval at 10–15 minutes.** Hammering every few seconds earns you rate limits, not capacity.
- **Run it somewhere always-on** — the AMD micro you launched in step 04 is the natural home.
- **Log every attempt** so you can tell "still no capacity" from "my script broke three days ago".
- **Don't pay for "priority".** Third-party "capacity sniper" services want your Oracle credentials. Never hand those over.

06 First 15 minutes on the box

SSH in

```
ssh -i ~/.ssh/your_key ubuntu@<public-ip>   # Ubuntu images: user "ubuntu"
                                            # Oracle Linux images: user "opc"
```

Add swap before anything else (mandatory on 1 GB)

1 GB of RAM with no swap means the kernel OOM-kills your process the first time apt, pip or a build spikes memory. 2 GB of swap makes the box livable:

```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Open ports in TWO places (the gotcha that burns everyone)

Traffic to an Oracle instance passes two firewalls, and both default to closed:

- **The VCN security list** (cloud console → your VCN → Security Lists): add an ingress rule per port — source `0.0.0.0/0`, TCP, destination port 80/443/whatever your app uses. Port 22 is open by default.
- **The OS firewall on the box itself:** Oracle's Ubuntu images ship with restrictive iptables rules baked in. Opening a port in the console does nothing until the OS allows it too. If a port that "should" work doesn't, check `sudo iptables -L INPUT` — that's almost always the culprit.

```
# Ubuntu: allow HTTP/HTTPS at the OS level
sudo iptables -I INPUT 6 -m state --state NEW -p tcp --dport 80 -j ACCEPT
sudo iptables -I INPUT 6 -m state --state NEW -p tcp --dport 443 -j ACCEPT
sudo netfilter-persistent save
```

07 What each box can actually run

BOX	GOOD FOR	DON'T BOTHER
AMD micro (1 GB)	nginx/Caddy static sites, reverse proxy, a Telegram/WhatsApp bot, cron workers, uptime monitors, small Flask/FastAPI/Node services, SQLite apps	LLM inference, video work, heavy Node builds (build elsewhere, copy artifacts), multi-container Docker stacks, Postgres under real load
ARM 2 OCPU / 12 GB	A legitimate small production server: app + Postgres + Redis, CI runners, multiple services behind nginx, small self-hosted tools (n8n, Uptime Kuma, Ghost…)	GPU work of any kind; anything needing x86-only binaries (it's aarch64 — most things ship ARM builds now, some don't)

The pairing covers a lot of ground for a pre-revenue founder: the micro is your always-on utility box today, the ARM machine is your real server once the script catches one.

08 The rules that keep it free

- ☐ Never upgrade to Pay As You Go. This is the only door a bill can walk through. Keep it shut.
- ☐ Stay inside the allowance table in section 02 — free-tier limits fail requests instead of billing you, which is exactly what you want.
- ☐ Only Always Free-tagged shapes and services. The console labels them.
- ☐ Don't terminate an ARM instance casually — after the June 2026 cut you may not get the same size back.
- ☐ Never give your Oracle credentials to third-party "get you a server" tools.
- ☐ Idle ARM instances can be reclaimed: Oracle may stop Always Free A1 instances with sustained very low utilization. A real workload (or a modest heartbeat job) keeps yours out of that bucket.

09 The checklist, top to bottom

- ☐ Sign up at signup.oraclecloud.com — home region = closest to your users (permanent).
- ☐ Card verification: small hold, released. Never charged unless you upgrade.
- ☐ Create VCN + public subnet. Generate an SSH key.
- ☐ Launch an AMD micro (VM.Standard.E2.1.Micro) today — instant in most regions.
- ☐ SSH in, add 2 GB swap, update packages.
- ☐ Open your ports in the security list AND in iptables on the box.
- ☐ Start the ARM retry script on the micro: every ~12 min, smallest shape first, log attempts.

- [ ] When the ARM box lands: move real workloads there, keep the micro as monitor/proxy.
- [ ] Never add a payment upgrade. Ever.

10 References

- Oracle Cloud Free Tier — `oracle.com/cloud/free`
- Always Free Resources (the canonical allowance list) — `docs.oracle.com/en-us/iaas/Content/FreeTier/freetier_topic-Always_Free_Resources.htm`
- Free Tier overview: trial vs Always Free, home region rules — `docs.oracle.com/en-us/iaas/Content/FreeTier/freetier.htm`
- InfoQ on the June 2026 ARM reduction (4/24 → 2/12) — `infoq.com/news/2026/07/oracle-cloud-free-tier-limits`
